PATENT ABSTRACTS OF JAPAN

(11)Publication number : 2000-211198
(43)Date of publication of application : 02.08.2000

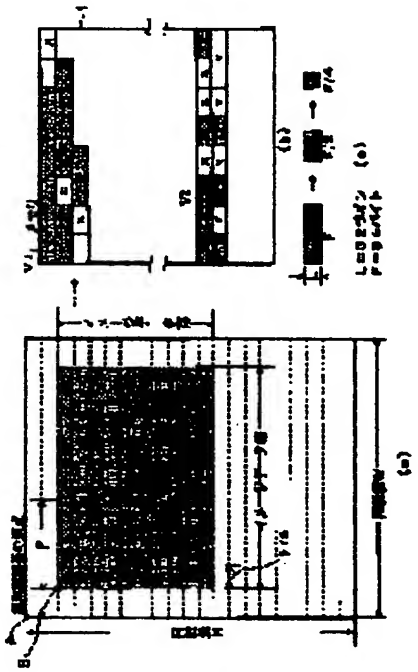
(51)Int.Cl. B41J 5/30
G06F 3/12

(21)Application number : 11-019386
(22)Date of filing : 28.01.1999

(71)Applicant : OKI DATA CORP
OKI DATA SYSTEMS CO LTD
(72)Inventor : SHIMAOKA AKIRA

(54) METHOD FOR MANAGING IMAGE DATA

(57)Abstract:
PROBLEM TO BE SOLVED: To effectively utilize a memory area by obtaining memory areas continuous on a memory for every block of a second size reduced from a size of the block when it is impossible to obtain a continuous memory area on the memory for storing blocks of a first size.
SOLUTION: A block consisting of a plurality of lines of a first size F is set. Image data is stored to memory areas V1 continuous on a memory 4 for every block. If the image data cannot be stored, a process is repeated whereby a size of the block is reduced to a second size F/2 and the image data is stored for every block of the second size. For example, a process of securing the memory data of 64 bytes by 32 lines, namely, approximately 2 kilo bytes as one unit is repeated. If the process is impossible, the memory of (32 bytes ×32 lines), i.e., one kilo bytes is obtained in the middle of the process. Accordingly, the image data can be stored integrally together as much as possible in the memory and a manage amount of data can be minimized.



LEGAL STATUS

[Date of request for examination] 20.01.2003

[Date of sending the examiner's decision of rejection]

[Kind of final disposal of application other than the examiner's decision of rejection or application converted registration]

[Date of final disposal for application]

[Patent number]

[Date of registration]

[Number of appeal against examiner's decision of rejection]

[Date of requesting appeal against examiner's decision of rejection]

[Date of extinction of right]

(19)日本国特許庁（ＪＰ）　(12)**公開特許公報（Ａ）**　(11)特許出願公開番号
特開2000－211198
（P2000－211198A）
(43)公開日　平成12年８月２日(2000.8.2)

(51)Int.Cl.⁷	識別記号	ＦＩ		テーマコード（参考）
Ｂ４１Ｊ　5/30		Ｂ４１Ｊ　5/30	Ｚ	２Ｃ０８７
Ｇ０６Ｆ　3/12		Ｇ０６Ｆ　3/12	Ｂ	５Ｂ０２１
				９Ａ００１

審査請求　未請求　請求項の数４　ＯＬ　（全 15 頁）

(21)出願番号　特願平11－19386

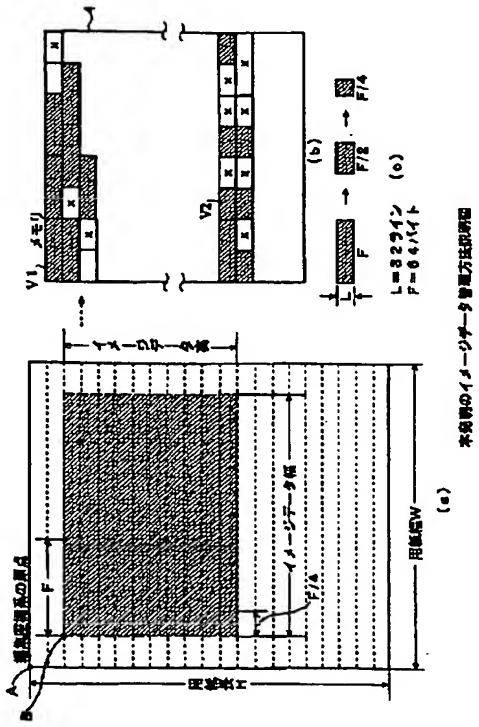
(22)出願日　平成11年１月28日(1999.1.28)

(71)出願人　591044164
株式会社沖データ
東京都港区芝浦四丁目11番地22号
(71)出願人　594202361
株式会社沖データシステムズ
福島県福島市庄野字立田１番地１
(72)発明者　嶋岡　章
福島県福島市庄野字立田１番地１　株式会社沖データシステムズ内
(74)代理人　100082050
弁理士　佐藤　幸男　（外１名）

最終頁に続く

(54)【発明の名称】　イメージデータの管理方法

(57)【要約】
【解決手段】　上位装置から受信したイメージデータは、予め第１のサイズＦの２以上のラインから成るブロックに分割する。各ブロックには印刷されるページ中の印刷位置情報を含む管理情報を用意する。このサイズのブロック毎にメモリ４への格納を繰り返し、記憶領域が見当たらないときは、ブロックのサイズを２分の１にして記憶領域を取得し、格納動作を繰り返す。こうして１ページ分のイメージデータをメモリ４に記憶し、印刷動作を実行する。
【効果】　１ラスタライン毎でなく、所定のサイズの２以上のラインから成るブロックに分割してイメージデータをメモリ４に記憶するので、全体として管理領域のデータ量を減少させ、メモリの使用効率が向上する。



1

【特許請求の範囲】

【請求項１】　メモリ上に１ページ分のイメージデータを記憶して印刷動作を実行する場合に、
前記イメージデータを一定の第１のサイズの２以上のラインから成るブロックに分割し、
前記各ブロックに対して、前記ページ中の印刷位置情報を含む管理情報を用意して、
この管理情報を前記メモリに格納するとともに、
前記第１のサイズのブロック毎に、前記イメージデータを前記メモリ上の連続した記憶領域を取得して格納し、
前記１ページ分のイメージデータ全体を記憶するまで、その動作を繰り返し、
前記第１のサイズのブロックのイメージデータを格納するための、前記メモリ上の連続した記憶領域を取得できなくなったときは、前記ブロックのサイズを縮小して第２のサイズとし、
前記第２のサイズのブロック毎に、前記イメージデータを前記メモリ上の連続した記憶領域を取得して格納し、
前記１ページ分のイメージデータ全体を記憶するまで、その動作を繰り返すことを特徴とするイメージデータの管理方法。

【請求項２】　請求項１に記載のイメージデータの管理方法において、
前記第２のサイズのブロックのイメージデータを格納するための、前記メモリ上の連続した記憶領域を取得できなくなったときは、さらに、前記ブロックのサイズを縮小して第３のサイズとし、
前記第３のサイズのブロック毎に、前記イメージデータを前記メモリ上の連続した記憶領域を取得して格納し、
前記１ページ分のイメージデータ全体を記憶するまで、その動作を繰り返すことを特徴とするイメージデータの管理方法。

【請求項３】　請求項１に記載のイメージデータの管理方法において、
前記各ブロックのイメージデータをメモリに格納する処理を一時中断して、
前記メモリから各ブロックのイメージデータをブロック毎に読み出して圧縮し、メモリに再格納することを特徴とするイメージデータの管理方法。

【請求項４】　請求項１に記載のイメージデータの管理方法において、
前記各ブロックのイメージデータをメモリに格納する処理を一時中断して、
前記メモリから各ブロックのイメージデータを読み出し、複数のブロックのイメージデータを結合してから圧縮して、メモリに再格納することを特徴とするイメージデータの管理方法。

【発明の詳細な説明】

【０００１】

【発明の属する技術分野】本発明は、ページプリンタにおけるイメージデータの管理方法に関する。

2

【０００２】

【従来の技術】ページプリンタは、上位装置、例えばパーソナルコンピュータ等から印刷データを受信して印刷をする。通常、ページプリンタでは、装置内部のメモリに１ページ分のデータを上位装置から受信した後、イメージデータへの変換処理を行う。１ページ分の印刷データを編集し、その変換処理が完了すると、印刷動作を開始する。印刷動作の開始と共にメモリ上に保持したイメージデータを順番に読み出し、ラスタバッファに展開する。ラスタバッファには、２値化されたイメージデータが一時格納され、プリントエンジンに転送される。

【０００３】こうしたページプリンタの印刷データ記述言語としては、ＨＰ－ＰＣＬ言語が挙げられる。この記述言語によれば、印刷データをラスタライン単位で上位装置からページプリンタに転送する。ラスタラインというのは、主走査方向即ち水平方向の長さが任意の長さであって、副走査方向即ち垂直方向の高さが１ドットの、ビットストリーム状データのことをいう。メモリには、このラスタライン単位でイメージデータが書き込まれて記憶される。印刷データがＬ本のラスタラインから成り、水平方向の長さＫが全て同一の場合には、長さがＫで高さがＬの矩形データとして、一括してメモリに書き込まれる場合もある。

【０００４】

【発明が解決しようとする課題】ところで、上記のような従来の技術には次のような解決すべき課題があった。メモリ上に１ラスタラインのイメージデータを記憶するためには、その印刷位置、長さ、高さ、解像度等の情報を含む管理情報をメモリに共に記憶する必要がある。解像度が６００ｄｐｉ（ドット／インチ）の印刷装置でＡ４判の用紙全体にイメージデータを印刷する場合、６０００ラスタライン以上のイメージデータがメモリに格納される。これらのデータを個別に管理する場合、１ラインで２０バイト程度の管理情報が必要になるから、全体で約１２０キロバイトの管理情報記憶領域が必要になる。

【０００５】標準的なページプリンタでは、２メガバイトのＲＡＭ（ランダム・アクセス・メモリ）が実装されている。この２メガバイトのメモリ空間のうち０．５メガバイト分は、ラスタバッファや受信バッファ、システムワーク等のために使用される。従って、運用状態で使用可能なメモリ空間は１．５メガバイト程度になる。このことを考慮すると、上記のような管理情報を記憶するために無視できない程度の記憶領域を消費するという問題があった。記憶領域の消費量を低減するためにデータを圧縮することも考えられる。データの圧縮は、メモリに１ラスタラインのデータを書き込む直前に、そのつど行われる。しかしながら、ラスタライン単位での圧縮では、圧縮効率が低いという問題がある。

【０００６】水平方向の長さがＫで高さがＬの矩形データとして、一括してメモリに書き込まれる場合には、管理情報も少なくなり、圧縮効率も高まる。しかしながら、このようなイメージデータの生成は上位装置のプリンタドライバに依存する。プリンタドライバの生成するデータは、上位装置上で動作するアプリケーション自体のデータ管理方法や、プリンタドライバとのデータ受け渡し方法に依存する。従って、プリンタ側では、印刷データの形式指定をすることができないから、プリンタ側でメモリを節約するための制御を自発的に行うことかできない。

【０００７】

【課題を解決するための手段】本発明は以上の点を解決するため次の構成を採用する。

〈構成１〉メモリ上に１ページ分のイメージデータを記憶して印刷動作を実行する場合に、上記イメージデータを一定の第１のサイズの２以上のラインから成るブロックに分割し、上記各ブロックに対して、上記ページ中の印刷位置情報を含む管理情報を用意して、この管理情報を上記メモリに格納するとともに、上記第１のサイズのブロック毎に、上記イメージデータを上記メモリ上の連続した記憶領域を取得して格納し、上記１ページ分のイメージデータ全体を記憶するまで、その動作を繰り返し、上記第１のサイズのブロックのイメージデータを格納するための、上記メモリ上の連続した記憶領域を取得できなくなったときは、上記ブロックのサイズを縮小して第２のサイズとし、上記第２のサイズのブロック毎に、上記イメージデータを上記メモリ上の連続した記憶領域を取得して格納し、上記１ページ分のイメージデータ全体を記憶するまで、その動作を繰り返すことを特徴とするイメージデータの管理方法。

【０００８】〈構成２〉構成１に記載のイメージデータの管理方法において、上記第２のサイズのブロックのイメージデータを格納するための、上記メモリ上の連続した記憶領域を取得できなくなったときは、さらに、上記ブロックのサイズを縮小して第３のサイズとし、上記第３のサイズのブロック毎に、上記イメージデータを上記メモリ上の連続した記憶領域を取得して格納し、上記１ページ分のイメージデータ全体を記憶するまで、その動作を繰り返すことを特徴とするイメージデータの管理方法。

【０００９】〈構成３〉構成１に記載のイメージデータの管理方法において、上記各ブロックのイメージデータをメモリに格納する処理を一時中断して、上記メモリから各ブロックのイメージデータをブロック毎に読み出して圧縮し、メモリに再格納することを特徴とするイメージデータの管理方法。

【００１０】〈構成４〉構成１に記載のイメージデータの管理方法において、上記各ブロックのイメージデータをメモリに格納する処理を一時中断して、上記メモリから各ブロックのイメージデータを読み出し、複数のブロックのイメージデータを結合してから圧縮して、メモリに再格納することを特徴とするイメージデータの管理方法。

【００１１】

【発明の実施の形態】以下、本発明の実施の形態を具体例を用いて説明する。

〈具体例１〉図１は、本発明のイメージデータ管理方法の説明図である。この図を用いて、まず本発明の概略を説明する。図の（ａ）には、印刷する用紙のアウトラインと、ここに印刷されるイメージデータ（ハッチングを付した部分）の関係を示した。印刷される用紙は、用紙幅がＷ、用紙長がＨである。編集座標系の原点はＡである。印刷されるイメージデータは、ここでは図の破線で囲んだブロック単位で取り扱う。

【００１２】図の左上端にあるブロックは、用紙幅Ｗの方向のサイズがＦバイト、用紙長Ｈの方向のサイズがＬドット、即ち長さＦでＬ本のラインにより構成される。この左上端にあるブロックのイメージデータは、その左上隅の点Ｂの座標を印刷位置情報とする。図示しない管理情報には、このような印刷位置情報が含められる。図の（ｂ）に、メモリ４のイメージデータを格納するための記憶領域を示した。管理情報は同じメモリ４の別の領域に記憶されているものとする。メモリ４には、イメージデータがブロック毎に記憶される。

【００１３】このメモリ４は、図（ｂ）の左上から右下に向かって、順にアドレスが設定されているものとする。（ｂ）に示すＶ１は、（ａ）の左上端にあるブロックのイメージデータを格納した記憶領域である。その後も同一サイズの記憶領域に、順に対応するブロックのイメージデータが記憶されていく。一方、メモリ４上には、イメージデータ以外に、例えばダウンロードフォントといった参照データも記憶される。イメージデータを記憶した領域には、１ページの印刷が終わると次のページのイメージデータが上書きされる。しかし、ダウンロードフォントのような参照データは、例えば全てのページの印刷が終了するまで、保持しておかなければならない。これらの参照データがメモリ中に散在すると、必ずしも連続した領域に上記のサイズのブロックのイメージデータを格納することができなくなる。

【００１４】そこで、この場合には、ブロックのサイズを、最初のＦから２分のＦにして、記憶領域を探す。図の（ｃ）には、このように、ブロックのサイズを縮小する例を示した。ライン数Ｌはそのままである。こうして、メモリ４上にサイズが２分のＦのブロックのイメージデータを格納できるような連続した記憶領域を取得できれば、そのブロックを格納する。サイズが２分のＦのイメージデータを格納した最初の記憶領域を、図中Ｖ２とした。メモリ４上には、印刷開始前に、少なくとも１ページ分のイメージデータが記憶される必要がある。従

って、１ページ分のイメージデータを格納するまで、このような処理を繰り返す。もし、２分のＦに縮小したブロックでも連続した記憶領域を取得できない場合には、４分のＦのサイズのブロックとし、同様の処理を繰り返す。

【００１５】（ｃ）には、例えば、図に示すように、ブロックの第１のサイズＦを６４バイト、ライン数を３２ラインとしたものを示した。このブロックを縮小する場合、第２のサイズを３２バイト、第３のサイズを１６バイトというように選定した。２分の１ずつ縮小したのは、コンピュータの処理単位を考慮したためである。管理データは、このようなブロック単位で生成され、メモリの所定の場所に記憶される。

【００１６】いずれの場合においても、１ブロックのデータ量が１ラスタラインのデータ量よりも十分に大きければ、従来のように１ラスタライン分ずつ管理データを生成してメモリ４に記憶していく場合に比べて、管理データを減少させることが可能になる。従って、全体として、管理データを記憶するべき記憶領域の節約が可能になる。以上が本発明の方法の概略である。

【００１７】図２には、用紙とイメージデータの関係説明図を示す。（ａ）は、図１に示した通りの用紙で、ここには既に説明した用紙幅W方向のサイズがＦ、用紙長Ｈ方向のサイズがＬの、１つのブロックを示した。

（ｂ）には、従来の管理情報と、本発明の管理情報との関係を示す。

【００１８】図に示すように、従来は１本のラスタラインＲに対し、それぞれ管理情報Ｄを生成してメモリに記憶するようにしていた。本発明では、２本以上のラインから構成されたブロックＶＶに対し、管理情報ＤＤを付加する。

【００１９】例えば、長さ６００バイトのラスタラインには２０バイト程度の管理情報が付加される。一方、上記ブロックのサイズＦを６２バイトとし、３２ライン分について、管理情報ＤＤを付加するとすれば、その管理情報の記憶領域を次のように節約することができる。例えば１ページ6000ラスタラインのイメージデータとすれば、従来、6000個の管理情報の記憶領域として１２０キロバイトを使用していたが、この発明では例えば1800個の管理情報の記憶領域として３６キロバイトを用意すればよい。

【００２０】図３には、管理情報の構造説明図を示す。上記のような管理情報は、例えばこの図に示すように、２バイトのチェインポインタ(Chain pointer)と、１バイトのフォーマットインフォメーション(Formatinformation)と、１バイトのプリントポジション(Print position)Ｘと、１バイトのプリントポジション(Print position)Ｙと、１バイトの高さ(Height)、１バイトの幅(Width)、合わせて１バイトの解像度(re)、２バイトのイメージデータアドレス(Image data Address)により構成される。

【００２１】この例では、これらの管理情報のすぐ後ろに６４×３２バイトのイメージデータが配置されている例を示す。チェインポインタは、メモリ開放のための管理ポインタで、他の管理情報との続き具合を示す例えば次の管理情報のアドレスが格納される。フォーマットインフォメーションは、データ種別、データバウンダリ、アドレスバウンダリ等の情報を含む部分である。プリントポジションＸ，Ｙは、それぞれ印刷位置のＸ座標とＹ座標を示す。例えばブロックの左上隅の座標を示す。

【００２２】座標やサイズは、例えば６００ｄｐｉの解像度の場合、６００分の１インチ単位で表示される。その後にイメージデータの高さと幅が表示される。また、解像度は、水平方向即ちＸ方向の解像度と垂直方向即ちＹ方向のものを示す。これらは、それぞれ３００ｄｐｉあるいは６００ｄｐｉといった形式で表示される。イメージデータアドレスは、この管理情報で管理されるイメージデータの先頭アドレスを表示する部分である。

【００２３】このように、管理情報は、ラスタライン１本分のものも、ブロック１個分のものも、表示する内容はほとんど変わらない。ラスタラインの場合には高さ情報が不要な程度である。故に、こうした管理情報の量をブロック単位で取り扱うことによって、メモリ消費量を減少させることができるのである。

【００２４】図４には、プリンタの主要部ブロック図を示す。図において、受信処理部１は、プリンタと他の上位装置等とのインタフェースである。この受信処理部１の受信したデータを格納するために、受信バッファ２が設けられる。また、これまで説明した要領でメモリ４上にイメージデータを記憶するために、編集処理部３が設けられている。メモリ４に格納されたイメージデータは編集処理部３を通じて展開処理部５に読み出され、展開されてラスタバッファ６に転送されるよう構成されている。なお、このラスタバッファ６に格納されたイメージデータは、図示しないプリントエンジンに転送されて印刷される。

【００２５】編集処理部３には、受信データ解析部１１、編集環境設定処理部１２、イメージデータ処理部１３、図形データ処理部１４、文字データ処理部１５、他データ処理部１６、ディスプレイリスト生成処理部１７、メモリ管理部１８、イメージデータディスプレイリスト生成処理部１９及びデータ圧縮処理部２０が設けられている。なお、データ圧縮処理部２０は、具体例２を実施する場合に利用される部分である。

【００２６】図５には、イメージデータ管理動作フローチャートを示す。上記のプリンタは、このフローチャートに従って動作する。まず、上位装置から転送される印刷データは受信処理部１が受信する。受信した印刷データの１ビットがイメージデータの１ドットに相当する場合と、印刷データの２ビットまたは４ビットあるいは８

ビットの複数ビットが、イメージデータ１ドットの階調値を指定する場合とがある。この例では、印刷データ１ビットがイメージデータ１ドットに相当する場合を説明する。印刷データが階調値を指定する場合には、例えばディザ法等により２値のマトリクスに展開処理を行い、全体が２値化された後のイメージデータを使用する。

【００２７】また、印刷データは一定の規則に従って圧縮されている場合もある。受信処理部１は、こうした印刷データを一旦受信バッファ２に記憶する。次に、編集処理部３の受信データ解析部１１が、受信バッファ２から印刷データを取り出し、その内容を解析する。印刷データは、従来どおりラスタライン単位で受信されるものとする。

【００２８】ここで、受信データ解析部１１は、各取り出した印刷データの印刷位置を算出する。印刷データの属性データには、印刷位置がＸ座標とＹ座標とで表示されている。印刷位置はそのラスタラインの先頭のビットを表示する位置である。水平方向の印刷位置は、例えば（Ｘ座標／１６）の商の整数部分とする。その端数は実印刷位置との差として保持し、展開時に補正する。垂直方向の印刷位置は（Ｙ座標／３２）の商の整数部分とする。端数はメモリへの格納位置補正に使用する。

【００２９】即ち、ここでは、水平方向のイメージデータの位置管理を１６ドット単位で行う。また、垂直方向のイメージデータの位置管理を３２ライン単位で行う。（Ｘ座標／１６）の商の整数部分をとると、実際に指定されている印刷位置よりも、“１”から“１５”ドットの範囲で水平方向に位置ずれを生じる。これは、記憶領域を取得するきっかけとなったブロックの最初のラインの印刷位置（Ｘ座標）を基準に補正すればよい。垂直方向も同様である。これらの計算結果を利用して、処理対象となったラスタラインを分割して得たイメージデータを、対応するブロックのために取得した記憶領域に順に位置補正をしながら格納していく。

【００３０】次に、ステップＳ２において、印刷データが圧縮されているかどうかを判断する。圧縮されていればステップＳ３に進み、圧縮アルゴリズムを使用して、印刷データを伸長する。この段階で全ての印刷データは２値の加工されていないイメージデータに変換された。ステップＳ４では、図４に示すイメージデータ処理部１３が、入力するイメージデータを既に説明した一定のサイズに分割する。即ち、１ラスタライン分のイメージデータの先頭から６４バイト分を分割して取り出す。

【００３１】次に、ステップＳ５において、記憶領域取得の要否を決定する。ここで記憶領域取得の要否を決定するのは、次のような理由による。受信データは、ラスタライン毎に受信される。そして、例えば最初のラスタラインを受信したとき、その先頭から６４バイト分を格納するために、メモリ上に所定の連続した記憶領域を取得する。このとき取得する記憶領域は、６４バイト×３２ライン分である。そのラスタラインのサイズ即ち図１のイメージデータ幅と表示した部分のサイズが６４バイト以上あれば、残りの部分についても６４バイトずつに分割し、それぞれ別々の記憶領域を６４バイト×３２ライン分ずつ取得する。

【００３２】次に新たなラスタラインを受信すると、その先頭の６４バイト分を、直前のラスタラインの先頭の６４バイト分のイメージデータに続けて格納する。この場合には、既に記憶領域が取得されているから、新たに記憶領域を取得する必要はない。この処理を３２ライン分続ければ、サイズが６４バイトで３２ライン分の例えば３個のブロックが、１ブロック分ずつメモリ上の連続した記憶領域に格納される。次のラスタラインが受信されたときは、新たに３個のブロックのための記憶領域を取得する。即ち、３２ラスタラインごとに新たな記憶領域の取得を行うので、ステップＳ５を設けたのである。

【００３３】図６には、こうした記憶領域取得動作のフローチャートを示す。まず、ステップＳ５１において、記憶領域が取得済みかどうかを判断する。ブロックの最初のラインを記憶する場合には、記憶領域の取得が必要であるからステップＳ５２に進み、「取得要」という判断結果をフラグ等で表示して、図５のステップＳ６に進む。ブロックの第２番目以下のラインを記憶する場合には、記憶領域が取得済みであるからステップＳ５１からステップＳ５３に進む。

【００３４】そして、ここで、これから格納しようとするイメージデータが、既に格納済みのイメージデータと水平方向の位置が重複するかどうかを判断する。イメージデータの先頭の位置が一致すれば、両方とも６４バイトのサイズであって、そのまま確保した記憶領域に格納できるから、ステップＳ５７で「取得不要」というフラグを立てて、ステップＳ５６に進む。もし、イメージデータの先頭位置が一致しないと、６４バイトのイメージデータであっても、先頭位置のシフト分だけ、サイズが実質的に大きくなる。即ち、確保しておいた６４バイト分の記憶領域に格納できない。

【００３５】そこで、その場合には、イメージでのサイズをシフト分だけ短くするように変更する。ステップＳ５５はそのための処理である。ステップＳ５４はステップＳ５７と同一の処理である。その後ステップＳ５６で、水平方向と垂直方向のデータシフト量を算出して、そのイメージデータを格納すべき位置を求め、次の処理に進む。

【００３６】再び図５に戻って、ステップＳ６では、上記の記憶領域取得要否の決定に基づいて、処理を分岐する。記憶領域の取得が必要でなければステップＳ１０に進み、図６のステップＳ５６の処理結果に従ってメモリの所定位置にイメージデータを格納する。次にステップＳ１１において、１ラスタライン分のイメージデータのうち分割した残りのイメージデータがあれば、そのサイ

ズをもとのサイズから６４バイト分差し引いたサイズに更新する。次のステップＳ１２において、全てのイメージデータの格納が完了したかどうかを判断する。１ページ分のイメージデータの処理を連続して行うためである。

【００３７】全てのイメージデータの格納が完了していなければステップＳ４に進み、残りのイメージデータについて、先頭から６４バイト分を分割し、これまで説明した処理を行う。残りのイメージデータが６４バイトに満たない場合にも、１ブロックのサイズは６４バイトとする。後から１ブロックのサイズが変更された場合には、それに従う。また、ステップＳ５において記憶領域の取得が必要と判断されたときは、ステップＳ６からステップＳ７に進み、メモリ４上に１ブロック分の記憶領域を取得する。この処理は、図４に示すメモリ管理部１８が行う。この場合、メモリ４上に１ブロック分のイメージデータを格納できる連続した記憶領域を探して、その先頭アドレスを得る。その後ステップＳ７からステップＳ８に進み、記憶領域を取得できたかどうかを判断し、取得できた場合にはステップＳ１０に進み、メモリにそのブロックの先頭のラインのイメージデータを格納する。

【００３８】ここで、連続した記憶領域が取得できない場合には、ステップＳ８からステップＳ９に進み、ブロックサイズを２分の１にする。即ち、６４バイト３２ライン分のブロックサイズを３２バイト３２ライン分にして、改めてメモリ４をサーチし、連続した記憶領域を取得する。以上のような処理を、ステップＳ１２で、１ラスタライン分のイメージデータの格納が完了するまで繰り返す。改ページ等のコマンドが検出されるまで、各ラスタラインのイメージデータを以上のような処理を繰り返すことにより、メモリに格納する。

【００３９】改ページコマンドを検出すると、メモリ４へのイメージデータの格納が終了する。その後、メモリ４中のイメージデータは、イメージデータディスプレイリスト生成処理部１９を通じて展開処理部５に読み出される。展開処理部５は、イメージデータを順次読み出してラスタバッファ６を通じて、図示しないプリントエンジンに転送する。上記のように、上位装置から１ラスタラインずつイメージデータを受信して、６４バイトのサイズに分割し、それをメモリに格納する作業を３２回繰り返して１ブロック分のイメージデータの格納を終えるが、そのデータは切れ目なく連続しており、その後は先頭アドレスとブロックの大きさを管理情報により表示しておけば、一括してメモリから読み出すことができる。

【００４０】上記のように一定のサイズ（第１のサイズ）の複数のラインから成るブロックを設定し、イメージデータをブロック毎にメモリ上の連続した記憶領域に格納し、それで格納できない場合にはブロックのサイズを縮小して、第２のサイズのブロック毎にイメージデータを格納するという処理を繰り返せば、メモリ中のデータの再配置処理を行うことなく、記憶領域を有効に利用することができる。即ち、既に説明したように、メモリ上には様々な保持しておかなければならないデータが散在することがあり、これらの存在によって、１ブロック分のイメージデータを連続して格納することができなくなることがある。この場合、これらを再配置させる処理を行えばよいが、印刷中にこのような処理を行えば、印刷速度が低下する。

【００４１】従って、図１（ａ）に示したように、例えば上記のように６４バイトで３２ライン、即ち約２キロバイトを１単位として記憶領域を確保する処理を繰り返し、これが不可能ならば途中から３２バイト×３２ライン即ち１キロバイトのメモリ取得を行うようにする。これでも連続した記憶領域の取得が不可能ならば、後半は１６バイト×３２ライン即ち０．５キロバイトの記憶領域取得を行うようにして１ページ分のイメージデータ格納処理を終える。こうすれば、可能な限りイメージデータをまとめてメモリに格納し、管理データ量を最小限にすることができる。

【００４２】ところで、上記のようなメモリに格納されたイメージデータは、印刷開始と同時にラスタバッファ６を通じてプリントエンジンに転送される。この場合のラスタバッファ６上へのイメージデータ展開速度は、用紙走行速度を考慮して定められる。即ち、一定時間内にラスタバッファ６への展開が完了しない場合には、いわゆるプリントオーバーランという現象が発生する。プリントエンジンは印刷用紙を搬送するが、必要なイメージデータの転送が間に合わず、白紙のまま印刷をしてしまう現象である。データの管理単位を１ラスタライン毎に設定しておくと、メモリ４から１ラスタライン毎にイメージデータが読み出されてラスタバッファ６へのデータ転送が行われる。

【００４３】これに対して、本発明のように、ブロック毎のイメージデータ管理を行うと、ラスタラインよりもデータ量が大きくなるように設定したブロック毎にイメージデータが読み出されて、ラスタバッファ６へのデータ転送が行われる。故に、イメージデータを展開し、ラスタバッファに転送する際の処理時間が短縮され、プリントオーバーランを防止する効果がある。

【００４４】例えば、いずれもＲＩＳＣ－ＣＰＵを使用した２台のプリンタＡ，Ｂであって、一方は８ｐｐｍ（ページ／分）、他方は１６ｐｐｍの印刷速度のものをシミュレーションにより比較して見る。プリンタＡは、制御クロック周波数２５メガヘルツ、Ｉキャッシュ（制御データ用キャッシュ）が４キロバイト、Ｄキャッシュ（データ用キャッシュ）が１キロバイトとする。プリンタＢは、制御クロック周波数３２メガヘルツ、Ｉキャッシュ３キロバイト、Ｄキャッシュ１キロバイトする。

【００４５】文字データを含むビットマップデータのラ

スタバッファへの展開に要する前処理時間は、プリンタＡ，Ｂ共に1016サイクルであった。いずれのプリンタもラスタバッファは１９２ラインの容量を持ち、用紙の走行速度はプリンタＡが２インチ／秒、プリンタＢは４インチ／秒であった。タスクの切り換え、割り込み、ビデオＤＭＡのバス占有等、展開処理以外の処理に要する時間を全体の処理の２０％とすると、プリンタの解像度が６００ｄｐｉの場合、１９２ラインの展開に許される時間は次のようになる。

プリンタＡ：（１９２÷（２×６００））×０．８＝０．１２８

プリンタＢ：（１９２÷（４×６００））×０．８＝０．０６４

【００４６】即ち、プリンタＡは１２８ミリ秒、プリンタＢは６４ミリ秒の展開可能時間となる。この時間内に展開処理を終了すれば、オーバーランにならない。イメージデータ展開の前処理が１ラスタライン当たり1016サイクルであって、１９２ラスタラインの前処理には195702サイクルが必要となる。従って、それぞれのクロック周波数を考慮した展開可能時間に占める前処理時間の割合は次のようになる。プリンタＡ：（195702サイクル÷２５メガヘルツ）÷１２８ミリ秒＝0.0611プリンタＢ：（195702サイクル÷３２メガヘルツ）÷６４ミリ秒＝0.0955

【００４７】即ち、プリンタＡは６．１１％、プリンタＢは９．５５％となる。即ち、ＣＰＵの搭載周波数と用紙の走行速度との関係によってある程度変化するが、この前処理に要する時間の割合が小さいほど、イメージデータと他のデータの展開に許される時間が増加する。従って、本発明のように、ブロック単位の処理を行い、ブロック毎に管理情報を用意することによって、展開に時間のかかるようなイメージデータを処理したとしても、プリントオーバーラン発生率の低下を図ることができる。

【００４８】〈具体例１の効果〉以上のように、ブロック単位でメモリ上に連続した記憶領域を取得し、その動作を繰り返し、第１のサイズのブロックを格納するためのメモリ上の連続した記憶領域を取得できなくなったとき、ブロックのサイズを縮小して第２のサイズとし、再び第２のサイズのブロック毎にメモリ上の連続した記憶領域を取得するといった方法で２以上のラインから成る一定のサイズのブロックでメモリを管理すると、管理情報の減少によってメモリ領域を有効に利用できる。更に、ラスタラインよりもデータ量が大きくなるように設定したブロックを用いるので、ラスタバッファへの転送処理のための時間を減少させ、プリントオーバーランの発生を低減できる。

【００４９】〈具体例２〉上記の例では、一定の容量のメモリに格納する管理データの量を減少させて、メモリを有効に活用する方法を示した。しかしながら、有限のメモリにいかに大量のイメージデータを記憶することができるかが、プリンタ等の製品の性能を評価する指標になっている。従って、イメージデータをメモリに効率よく格納するだけでなく、よく知られたデータ圧縮を併用することにより、上記の本発明を一層効果的に利用することができる。

【００５０】例えば、上記のように３段階のサイズを設定し、最初６４バイト、次は３２バイト、その次は１６バイトのサイズでブロックを設定して連続する領域を探しても、連続してイメージデータを記憶する領域が見当たらないような場合には、メモリフルという状態が発生する。１ブロックのデータ量を１ラスタラインより小さくしてしまっては、イメージデータをブロックにまとめた効果がない。そこで、この具体例では、メモリに格納されている全てのデータをブロック単位で順番に読み出し、それぞれよく知られたアルゴリズムを用いて圧縮する。これによって、メモリの記憶領域を増大させる。

【００５１】図７には、ファイル圧縮率の説明図を示す。ここでは、例えば様々なデータ形式のファイルを１２０種類ほど用意し、それぞれ２種類の圧縮方式を採用し、元のデータに対し圧縮後のデータ数（単位はバイト）がどの程度になったか、その圧縮率はどの程度かという結果を示している。圧縮元のデータ数は、全ファイル、幅６００バイト、高さ6000ラスタライン固定の3,840,000バイトである。例えば、ファイル番号が１番のファイルは、図に示すように、圧縮方式Ａでは、圧縮後のファイルサイズは252,221バイトであって、６．５６％の圧縮率となる。データの形式やファイルの形式や圧縮方式によって、圧縮率が様々な範囲に変化することがわかる。

【００５２】図８には、このようなファイル圧縮率に着目して、２種の圧縮方式を用いた場合の頻度表を図示した。例えば、図に示す圧縮方式Ａでは、圧縮率が５％未満のものが最も頻度が高く、圧縮率が１０～１５％未満のものが２７ファイルある。また、圧縮率が４５～５０％未満のものも１％存在する。圧縮方式Ｂでも類似した傾向が見られる。こうした結果から、平均的には圧縮前のメモリ消費量の３０％程度まで、ファイルサイズを圧縮することができるということが考えられる。

【００５３】従って、メモリに格納されたデータをメモリフルを検出した段階あるいはそれ以前の適当なタイミングで圧縮すれば、その時点で、残りのイメージを格納することができる記憶領域を確保できる。図４に示すデータ圧縮処理部２０は、このような圧縮処理を行うために用意された部分である。圧縮アルゴリズムについては、従来よく知られた任意の方法で良い。既に説明したように、２ＭバイトのＲＡＭを実装したページプリンタでは、運用時に使用可能な記憶領域は１．５Ｍバイト程度である。一方、Ａ４判の用紙全体に６００ｄｐｉで印刷をする場合、６０００ラスタライン以上のイメージデ

一タが必要で、そのデータ量は４Ｍバイト程度となる。上記の圧縮処理を実行すれば、実装メモリの増設が不要になる。

【００５４】図９は、具体例２の処理動作フローチャートである。このフローチャートは、具体例１のような処理を実行中に、メモリフル状態が発生したとき、実行される処理を示す。まず、ステップＳ１において、メモリがイメージデータを保持しているかどうかを判断する。メモリ中にイメージデータが存在していればステップＳ２に進み、メモリ中の全てのイメージデータをブロック毎に取り出し、順番に圧縮する。次に、ステップＳ３において、要求のあった記憶領域を取得できるかどうかの判断を行う。

【００５５】即ち、圧縮処理を終了した後にメモリの記憶領域に十分な空きが発生すると、具体例１の記憶領域取得処理を続けることができる。記憶領域が取得できればステップＳ５に進み、その記憶領域を取得して、具体例１のイメージデータ格納処理に進む。記憶領域を取得できなければ圧縮をしても効果がなかったと判断し、ステップＳ４において、メモリフルという表示等を行い、ページ編集を終了すればよい。

【００５６】〈具体例２の効果〉以上のように、任意のタイミングであるいはメモリフルが発生した場合に、メモリ中のデータ圧縮を行うことによって、メモリをより有効に使用することが可能になる。特に、メモリフルを検出した場合のみデータ圧縮を行うことによって、不必要なデータ圧縮処理による処理速度の低下を回避することができる。例えば従来のようにラスタライン単位で上位装置からデータの転送を受ける場合、データを圧縮するモードか圧縮しないモードかをデータの受信開始時に決定して、圧縮が必要ならば全てのデータについて圧縮をする。従って、処理速度の低下を招いていた。一方、本発明では、メモリフルになった時点でメモリ中のデータ圧縮をするから、メモリフルにならなければ、無駄な圧縮はしない。また、ブロック単位でデータを読み出して圧縮をするので、ラスタライン単位で読み出すよりも高速に処理できるという効果がある。

【００５７】〈具体例３〉上記の例では、メモリに格納されたイメージデータをブロック単位で管理していることに着目し、メモリからブロック単位でイメージデータを取り出し、これを順次圧縮し、メモリに再格納するという方法を採用した。しかしながら、各ブロックは最大２キロバイト程度であって、従来の１ラスタライン単位で管理されていたデータを圧縮するより圧縮率は改善できるが、それ以上の圧縮率は望めない。ところが、圧縮対象となるイメージデータのデータ数と圧縮効率の関係をシミュレーションすると、データ数が増加するほど圧縮率が向上し、圧縮対象データ数が８キロバイトを超える範囲でほぼ飽和することが判明した。

【００５８】図１０には、このデータサイズ別圧縮率比較説明図を示す。図に示すように、高さが１２８ライン、６４ライン、３２ラインのものであって、幅１６バイトのブロック、幅３２バイトのブロック、幅６４バイトのブロック、幅１２８バイトのブロック、幅２５６バイトのブロック、幅５１２バイトのブロックについて、それぞれデータ圧縮を行うと、その圧縮後のデータ数と圧縮率が様々な結果を示す。（ａ）は比較的圧縮率が低い場合の例で、（ｂ）は比較的圧縮率が高い場合の例である。

【００５９】この結果からみると、幅１２８バイト～幅５１２バイトのブロックサイズのものは、圧縮データの個々のサイズが８キロバイトを超えている。このような点を考慮すると、１ブロックのデータ数は大きい方が良いといえる。しかしながら、上記のようなメモリへのデータ格納のために連続した記憶領域を確保する処理を伴う場合には、ブロックを適切なサイズに選定することが好ましい。そこで、この具体例３では、データ圧縮の際、いくつかのブロックを読み出し、複数のブロックをまとめて圧縮し、メモリに再格納するという処理を行う。これによって、圧縮率を向上させている。

【００６０】図１１は、具体例３の動作説明図である。また、図１２は、この結合圧縮動作の説明図である。更に、図１３は、具体例３の動作フローチャートである。これらを用いて具体例３の動作を順に説明する。

【００６１】図１１（ａ）に示すように、これまで説明した通りの用紙のレイアウト中に、（１）－１～（８）－３及びＡ１～Ｃ５のブロックが存在し、これらが既に説明した要領でメモリ中に格納されているものとする。図の（１）－１のブロックの最初のラスタラインのイメージデータを受信すると、（１）－２、（１）－３のブロックのイメージデータについて、同時にブロック毎の記憶領域を取得し、既に説明した要領で、１ラスタライン受信する度に、順番に該当する記憶領域にイメージデータを格納する。このような処理が図の右側に示した（１）～（１１）まで繰り返される。

【００６２】なお、（９）では、ブロックのサイズが２分の１にされ、これまでと同様の要領でメモリへの格納処理を続ける。ここで、圧縮処理を行う場合には、（１）に示す３つのブロック即ち（１）－１のブロック、（１）－２のブロック、（１）－３のブロックをまとめてから圧縮を行う。また、図１１（ｂ）には、各ブロックのイメージデータのデータ構造を図示した。それぞれ、６４バイト×３２ライン分のデータが連続した構造をしている。上記の状態でメモリフルが発生したとする。ここで、つぎのようにしてデータの圧縮を行う。

【００６３】図１２には、この具体例３における結合圧縮動作を示す。図１２（ａ）に示すように、例えば図１１の（１）－１から（１）－３の３つのブロックのイメージデータを結合させ、連続する１つのデータとして圧縮処理をする。（１）－１と（１）－２は６４バイトの

サイズのブロックで、（１）－３は３２バイトのサイズのブロックである。これらを結合し、図１２に示すように、６４×２＋３２バイトのサイズで３２ラインのイメージデータから成るブロックを得る。即ち、サイズ１６０バイト３２ラインのブロックを一括して圧縮対象とする。図１２（ｂ）では、例えば、Ａ１～Ａ５のブロックを５個まとめる。それぞれ３２バイトのサイズのブロックであるため、これらを５つ結合し、サイズ１６０バイトで３２ラインのブロックとして圧縮を施す。こうすれば、既に説明をしたように、圧縮率がより高まる。

【００６４】図１３は、上記の処理を説明するための動作フローチャートである。この処理は、具体例２と同様に、メモリフルの状態が発生してから開始する。図に示すように、ステップＳ１において、未圧縮のイメージデータがあるかどうかをメモリ中で検索し、存在すればステップＳ２に進み、水平方向に隣接するデータがあるかどうかを判断する。即ち、図１２を用いて説明したように、水平方向に隣接するデータがある場合には、これらを全てまとめて結合し、１つのブロックとして圧縮を行うようにする。隣接データが存在する場合にはステップＳ３において、隣接するデータを結合させて圧縮する。存在しない場合はステップＳ４に進み、単独のメモリの圧縮を行う。以上の処理によって、圧縮率のより高いデータサイズにした上で圧縮を行う。

【００６５】以上のような処理は、印刷する用紙がランドスケープ形式の場合でも可能である。ランドスケープ形式とは、用紙を横に使用する形式で、これまで説明したのは、用紙を縦に使用するポートレイト形式である。図１４には、ランドスケープの場合の動作説明図を示す。ランドスケープの場合、例えばこの図に示すように、ブロックを用紙長Ｈの方向にとる。即ち、用紙長Ｈの方向にブロックの第１のサイズ即ち６４バイトの長さをとり、用紙幅Ｗの方向にラインを設定してイメージデータを処理する。その他の処理手順についてはこれまでの例と全く同様である。

【００６６】図１５には、ランドスケープの場合の圧縮処理説明図を示す。図の（ａ）に示すように、（１）－１～（８）－３までの第１のサイズのブロックと、Ａ１～Ｃ５までの第２のサイズのブロックのイメージデータをメモリ中に格納している。これが１ページ分のデータとする。ここで、ランドスケープのデータは、メモリから読み出されてラスタバッファ６に転送される場合、９０°回転させて、ポートレイト形式のデータに変換される。プリントエンジンはポートレイト形式のデータのみを受け付ける構造にされているからである。図１５（ｂ）には、例えば（１）－１～（８）－１のデータを回転させた後の、データを示す。この例では、（１）－１～（８）－１のデータを回転させてから結合し、これらをまとめて圧縮する。

【００６７】図１６には、回転後の結合圧縮動作説明図を示す。図１６（ａ）に示すように、例えば（１）－１の６４バイト３２ラインのブロックが回転されて、サイズ４バイトの５１２ラインのブロックに変換される。他の（２）－１～（８）－１のブロックも同様に回転される。そして、回転後の（１）－１～（８）－１の８個のブロックが結合される。こうして１６キロバイトの圧縮対象ブロックを生成する。

【００６８】（ｂ）は、Ａ１～Ｃ５のメモリ空間について、その回転後の結合圧縮を示す動作説明図である。この場合、Ａ１，Ｂ１，Ｃ１の３２ライン３２バイトのブロックを回転して４バイト２５６ラインのブロックに変換する。そして、これら３つのブロックを結合して、４バイト×３のサイズで２５６ラインのブロックを生成し、圧縮対象を３キロバイトとして圧縮する。このように、回転後、結合するという処理によって、ランドスケープタイプについても同様のデータ圧縮が可能になる。

【００６９】図１７には、具体例３の回転圧縮の場合の動作フローチャートを示す。この処理も、メモリフルの状態が発生した後で開始する。まず、ステップＳ１において、未圧縮イメージデータがあるかどうかを判断し、あればステップＳ２に進み、垂直方向に隣接データがあるかどうかを判断する。そして、垂直方向に隣接するデータがあればステップＳ３に進み、隣接する個々のメモリの回転を行う。更に、ステップＳ４に進み、図１５、１６で説明した要領で、回転後の複数のメモリを結合させて圧縮する。また、垂直方向に隣接するデータがなければステップＳ５に進み、メモリの回転を行って、回転後のメモリの圧縮を単独で行えばよい。

【００７０】図１８には、圧縮対象データの分割例説明図を示す。図には、５１２バイト１２８ラインのブロックと、これを１６バイトサイズで分割した場合、３２バイトサイズで分割した場合、６４バイトサイズで分割した場合、１２８バイトサイズで分割した場合、２５６バイトサイズで分割した場合を図解した。具体例１では６４バイトサイズ３２ラインのブロックを１単位として処理した。ところが、データの圧縮をする場合には、図１０を用いて説明したように、データ数の大きいものほど圧縮率が高い。そのデータ数を図示比較したものがこの図１８である。

【００７１】例えば６４バイト、高さ３２ラインのブロックで分割してイメージデータを管理し、その各ブロックをそのブロック単位で圧縮した場合には、低圧縮率のデータを圧縮する場合、８０．０%、高圧縮率のデータを圧縮する場合、２９．７５%程度までデータ圧縮ができる。

【００７２】一方、具体例３のように、また、隣接するブロックを結合させて一括して圧縮処理をすると、低圧縮率のデータを圧縮する場合には、７４．１８%、平均的なデータを圧縮する場合、２１．３１%まで圧縮することができる。圧縮後にメモリに約８０%の空きが生じ

17

るわけである。このような結果からブロックをどの程度結合するかを選定してその最適化を図ることができる。

【００７３】〈具体例３の効果〉以上のように、ブロック毎にイメージデータを管理し、メモリに格納した場合においても、隣接するブロックを結合して、一括して圧縮をすることにより、圧縮効率が向上し、メモリの使用効率を向上させることができる。

【図面の簡単な説明】

【図１】本発明のイメージデータ管理方法説明図である。

【図２】用紙とイメージデータの関係説明図である。

【図３】管理情報の構造説明図である。

【図４】プリンタの主要部ブロック図である。

【図５】イメージデータ管理動作フローチャートである。

【図６】記憶領域取得動作フローチャートである。

【図７】ファイル圧縮率の説明図（その１）である。

18

＊【図８】ファイル圧縮率の説明図（その２）である。

【図９】具体例２の動作フローチャートである。

【図１０】データサイズ別圧縮率比較説明図である。

【図１１】具体例３の動作説明図である。

【図１２】結合圧縮動作の説明図である。

【図１３】具体例３の動作フローチャートである。

【図１４】ランドスケープの場合の動作説明図である。

【図１５】ランドスケープの場合の圧縮処理説明図である。

【図１６】回転後の結合圧縮動作説明図である。

【図１７】具体例３の動作フローチャートである。

【図１８】圧縮対象データの分割例説明図である。

【符号の説明】

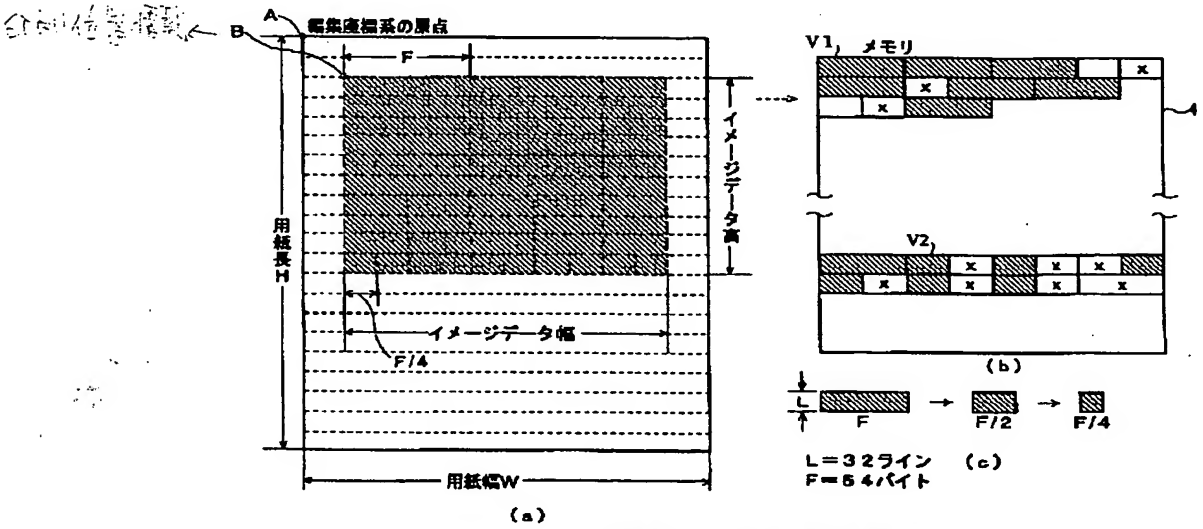
４　メモリ

Ｆ　第１のサイズ

Ｆ／２　第２のサイズ

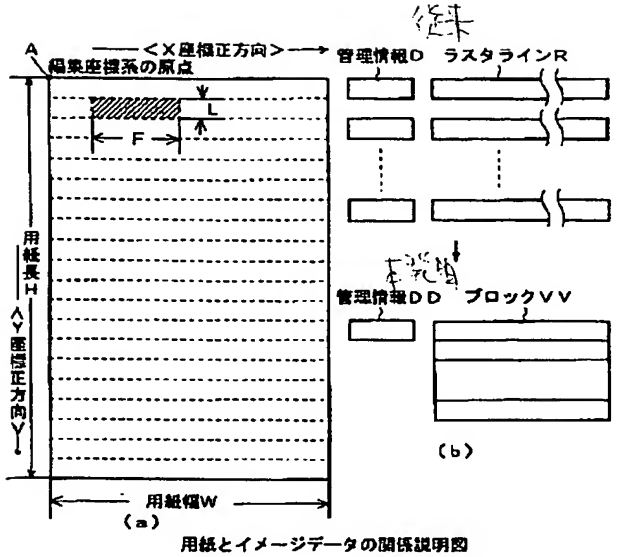
Ｖ１，Ｖ２　記憶領域

【図１】



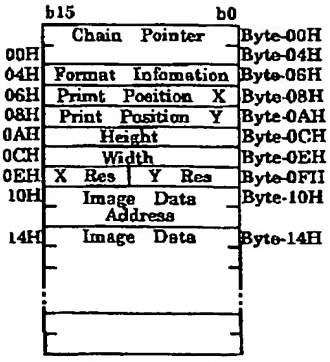
本発明のイメージデータ管理方法説明図

【図２】



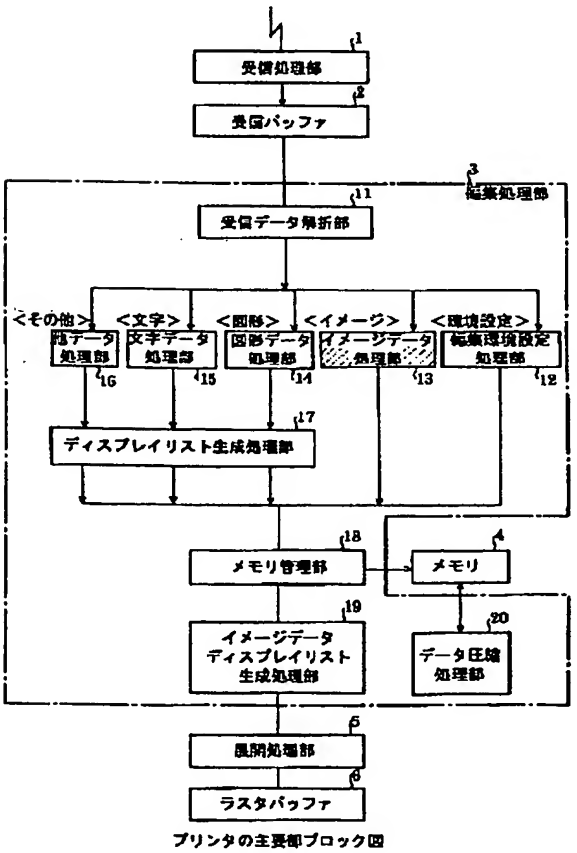
用紙とイメージデータの関係説明図

【図３】



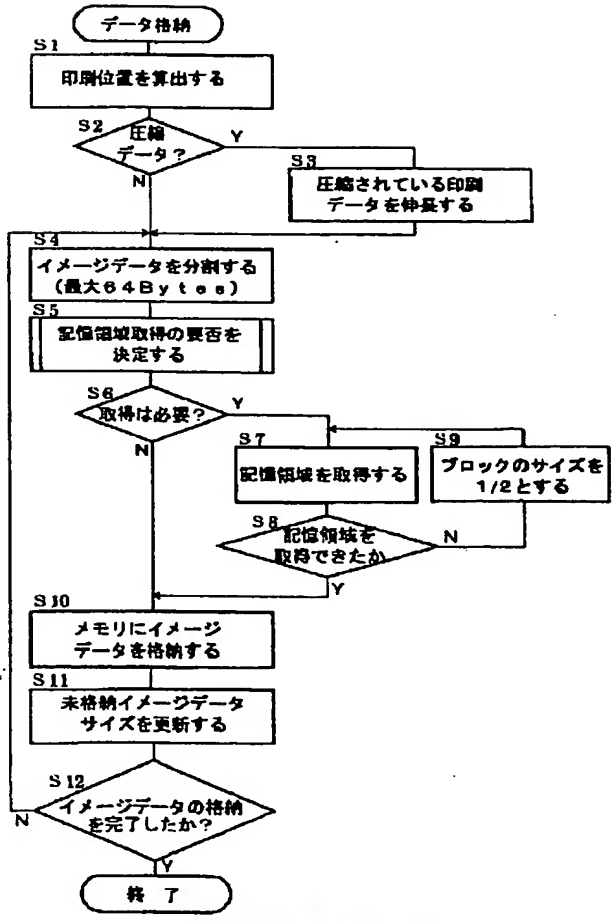
管理情報の構造説明図

【図４】



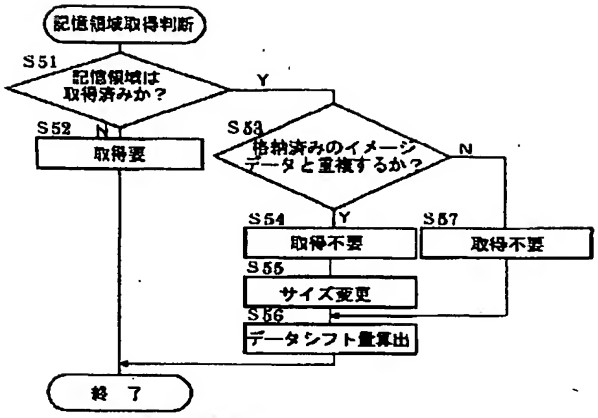
プリンタの主要部ブロック図

【図５】



イメージデータ管理動作フローチャート

【図６】



記憶領域取得動作フローチャート

【図７】

File No	圧縮後データ数		圧縮率	
	圧縮方式(A)	圧縮方式(B)	圧縮方式(A)	圧縮方式(B)
1	262,221	215,817	8.56%	5.62%
2	109,732	82,592	2.86%	2.15%
3	240,276	201,487	6.26%	5.25%
4	98,819	81,453	2.57%	2.12%
5	186,285	149,617	4.86%	3.90%
⋮				
36	127,386	142,627	3.32%	3.72%
37	437,542	509,460	11.39%	13.27%
38	468,571	548,749	12.20%	14.29%
39	350,102	919,723	9.12%	23.95%
40	327,532	845,617	8.53%	22.03%
⋮				
119	291,425	275,009	7.59%	7.16%
120	294,888	261,248	7.68%	6.80%

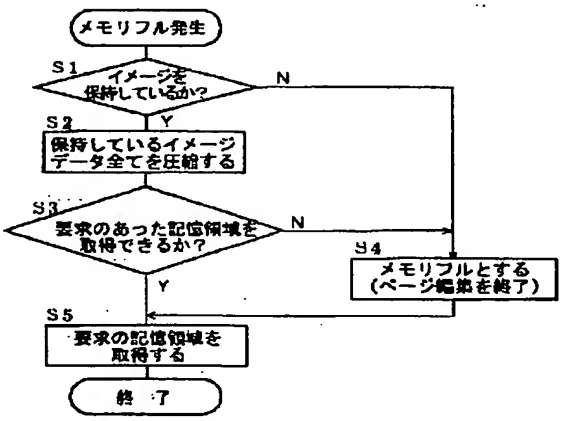
ファイル圧縮率の説明図（その１）

【図８】

圧縮率	圧縮方式(A)	圧縮方式(B)
～5%未満	33	25
5%～10%未満	39	27
10%～15%未満	27	29
15%～20%未満	8	15
20%～25%未満	3	11
25%～30%未満	3	2
30%～35%未満	2	4
35%～40%未満	1	0
40%～45%未満	1	2
45%～50%未満	1	1
50%～55%未満	1	2
55%～60%未満	0	0
60%～65%未満	0	1
65%～70%未満	0	0
70%～75%未満	1	0
75%～80%未満	0	0
80%～85%未満	0	1
85%～90%未満	0	0
90%～95%未満	0	0
95%～100%	0	0

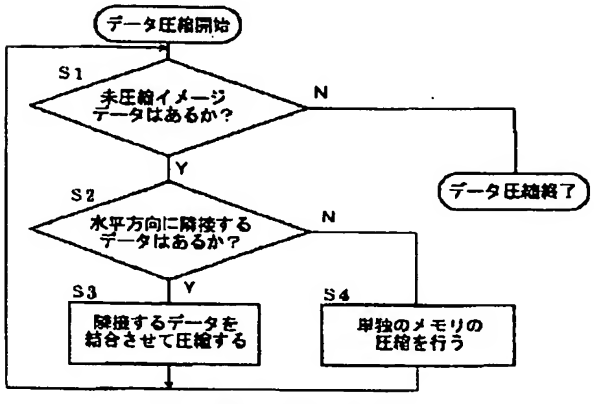
ファイル圧縮率の説明図（その２）

【図９】



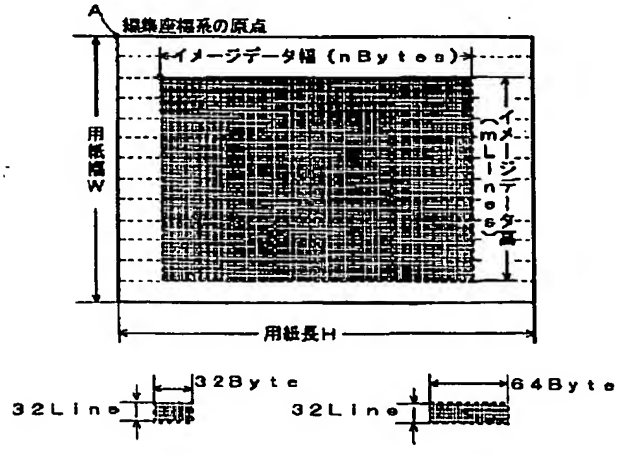
具体例２の動作フローチャート

【図１３】



具体例３の動作フローチャート

【図１４】



ランドスケープの場合の動作説明図

【図１０】

	圧縮前データ数	幅:16 Bytes データ数	圧縮率	幅:32 Bytes データ数	圧縮率	幅:64 Bytes データ数	圧縮率
高さ:128 Lines	65,536	64,955	99.13%	57,049	87.05%	52,779	80.53%
高さ: 64 Lines	32,768	32,483	99.13%	28,508	87.00%	26,284	80.21%
高さ: 32 Lines	16,384	16,059	98.02%	14,190	86.61%	13,112	80.03%

	圧縮前データ数	幅:128 Bytes データ数	圧縮率	幅:256 Bytes データ数	圧縮率	幅:512 Bytes データ数	圧縮率
高さ:128 Lines	65,536	50,435	76.96%	49,092	74.91%	48,362	73.79%
高さ: 64 Lines	32,768	25,127	76.68%	24,424	74.54%	24,049	73.39%
高さ: 32 Lines	16,384	12,608	76.95%	12,323	75.21%	12,153	74.18%

圧縮率低

（a）

	圧縮前データ数	幅:16 Bytes データ数	圧縮率	幅:32 Bytes データ数	圧縮率	幅:64 Bytes データ数	圧縮率
高さ:128 Lines	65,536	35,305	53.87%	26,093	39.82%	20,939	31.95%
高さ: 64 Lines	32,768	17,696	54.00%	13,012	39.71%	10,440	31.86%
高さ: 32 Lines	16,384	8,632	52.69%	6,228	38.01%	4,874	29.75%

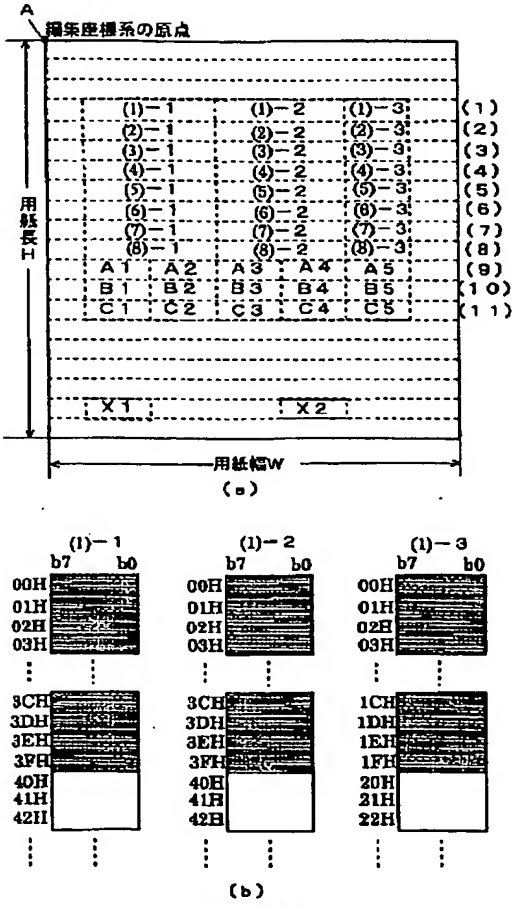
	圧縮前データ数	幅:128 Bytes データ数	圧縮率	幅:256 Bytes データ数	圧縮率	幅:512 Bytes データ数	圧縮率
高さ:128 Lines	65,536	18,149	27.69%	16,605	25.34%	15,687	23.94%
高さ: 64 Lines	32,768	8,985	27.42%	8,239	25.14%	7,766	23.70%
高さ: 32 Lines	16,384	4,125	26.18%	3,735	23.80%	3,492	21.31%

圧縮率高

（b）

データサイズ別圧縮率比較説明図

【図１１】

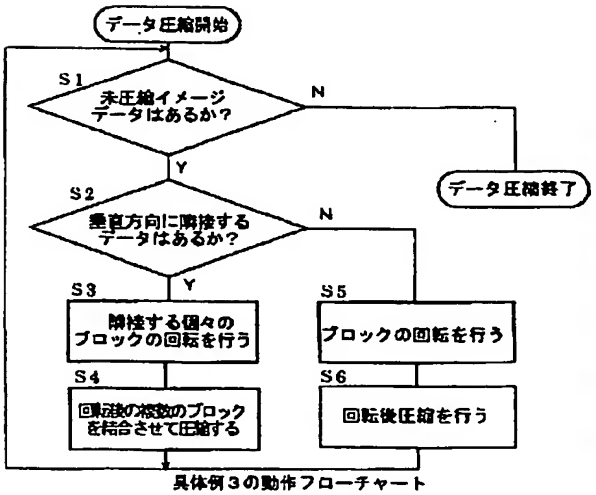


（a）

（b）

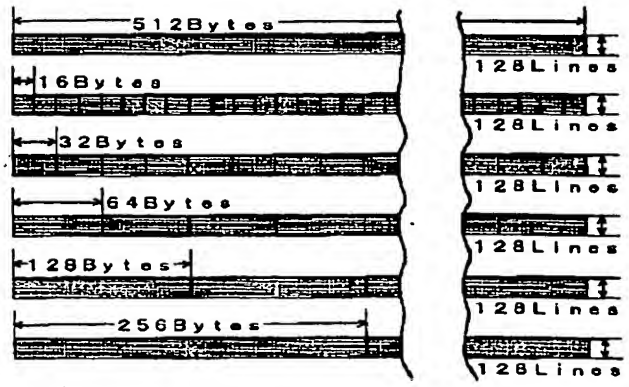
具体例３の動作説明図

【図１７】



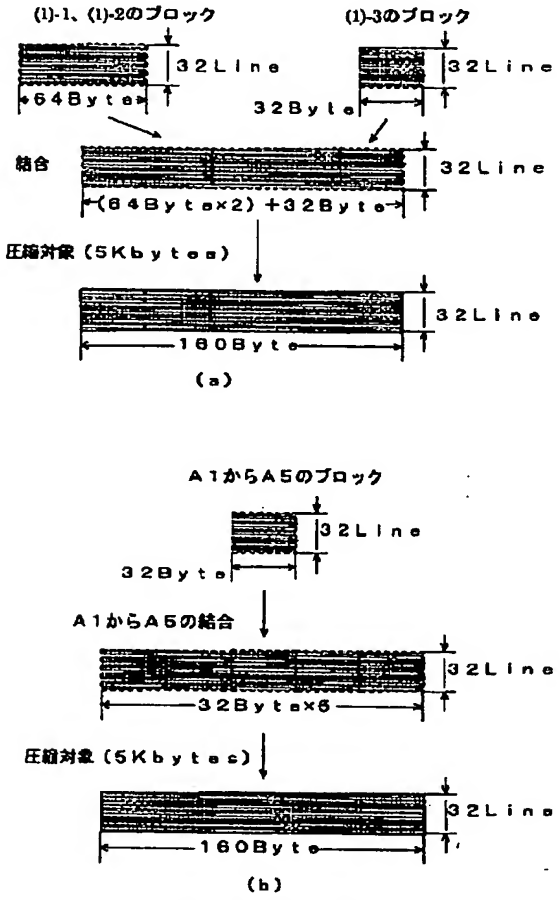
具体例３の動作フローチャート

【図１８】



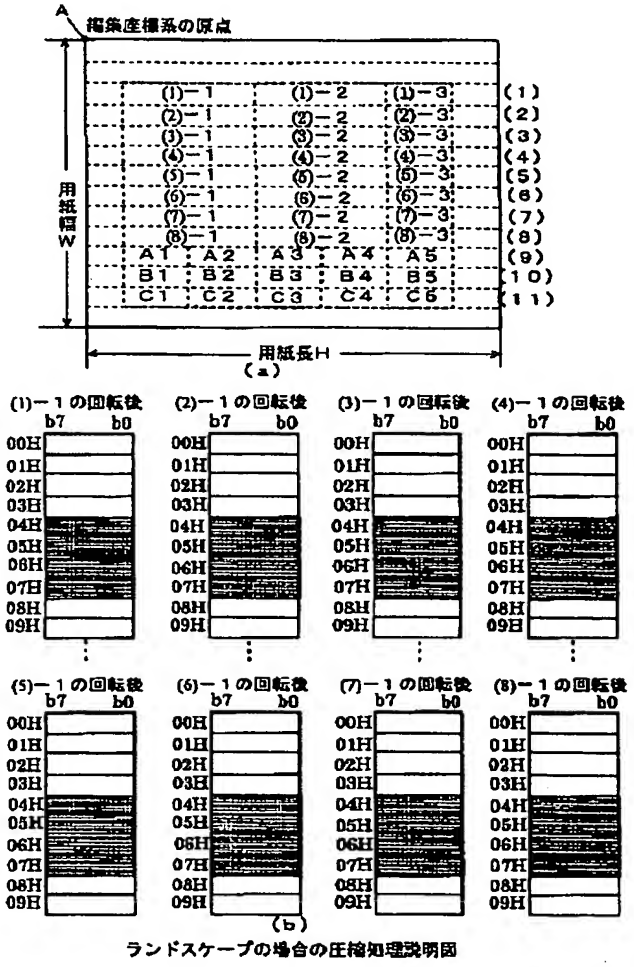
圧縮対象データの分割例説明図

【図１２】



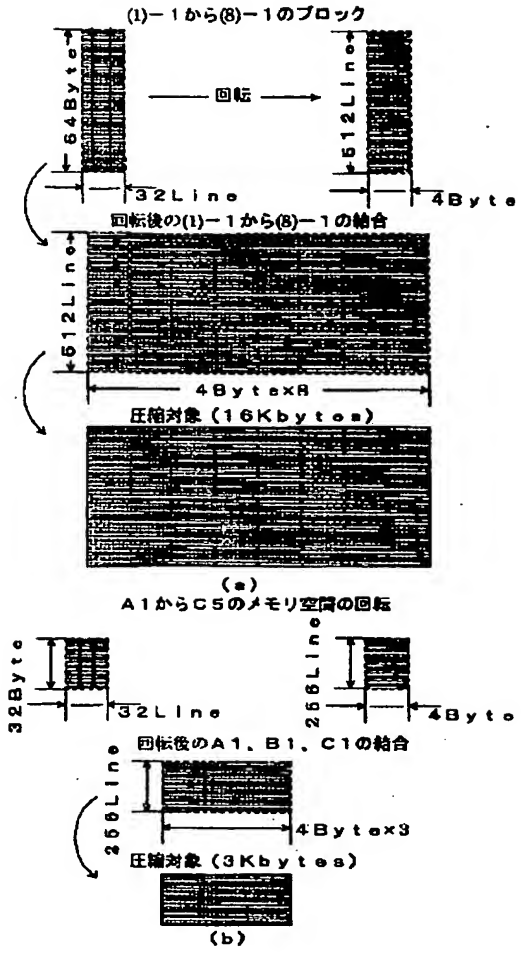
結合圧縮動作の説明図

【図１５】



ランドスケープの場合の圧縮処理説明図

【図１６】



フロントページの続き

Ｆターム(参考) 2C087 AB05 BC02 BC05 BC06 BD06
BD40 CA03
5B021 AA01 AA02 BB02 CC08 DD08
9A001 DD08 EE04 HH24 HH27 JJ35